Session 614

Advanced Java Client

Andreas Wendker
Manager, WebObjects Rapid Development

Introduction

- Java Client is a technology for creating distributed 3-tier desktop applications
  - Client displays user interface
  - Server accesses database
  - Database stores data
  - HTTP standard communication protocol
- Platform independent, localizable
- Designed for Intranet, not Internet

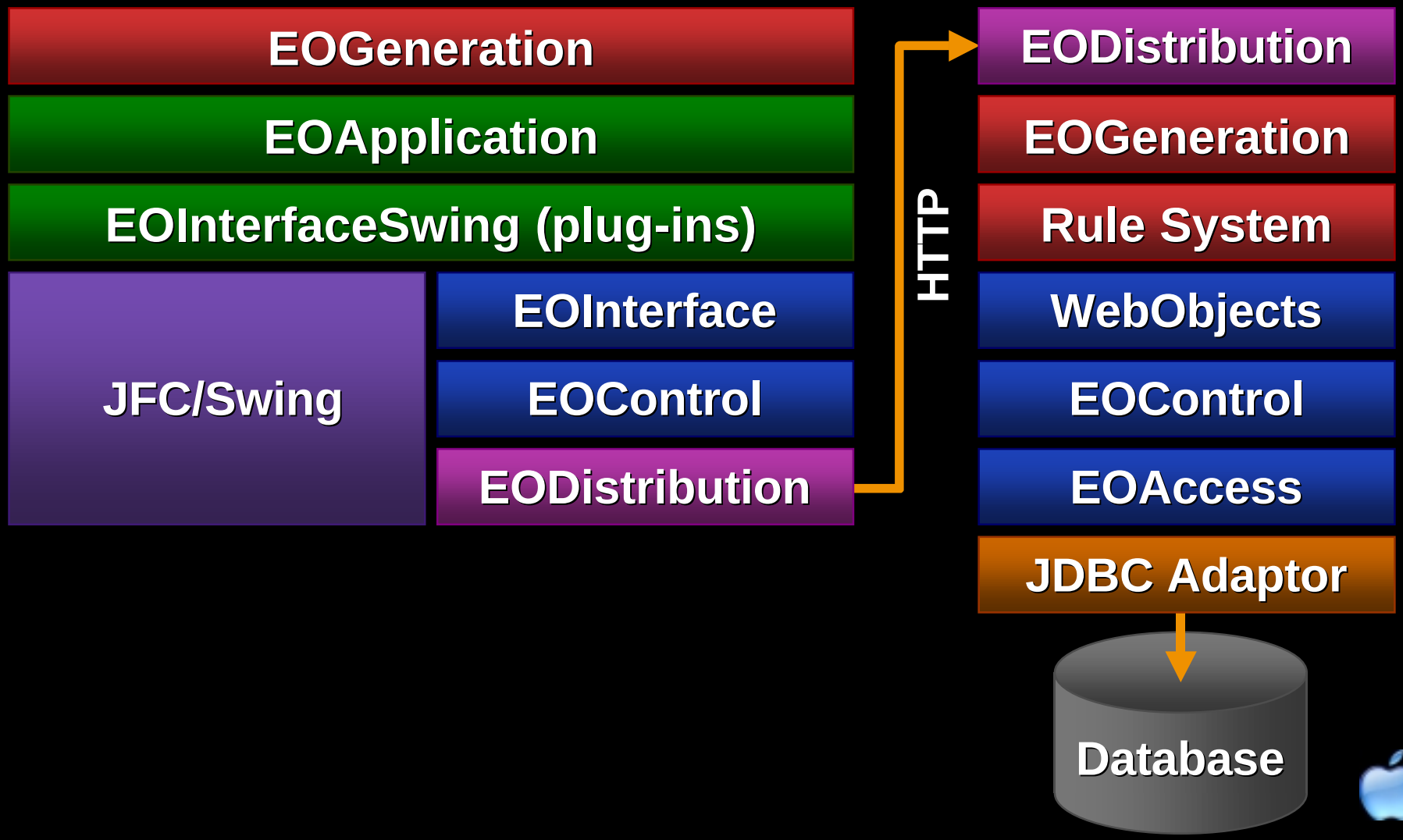
Introduction

- Java Client is incredibly powerful and well prepared for advanced implementations
- Some features are a little hidden, but you should really know about them

What You'll Learn

- What are the important features of the distribution layer?
- What should you know about deploying Java Client applications?
- How can Direct to Java Client's controller factory help you?

Application Architecture



Agenda

- Distribution Layer
  - Business Logic Partitioning
  - Distribution Layer Objects
  - Remote Method Invocations
  - Distribution Channels
  - Delegates
- Deployment
- Direct to Java Client: Controller Factory

Business Logic Partitioning

- Business logic on client and server can be identical or partitioned
  - Client side properties can be restricted or extended with derived attributes
  - Business logic implementations can be kept on the server
- Added security
- Performance optimization
  - Processing time
  - Transferred amount of data

Business Logic Partitioning

- Different server side and client side classes

Rentals.eomodeld

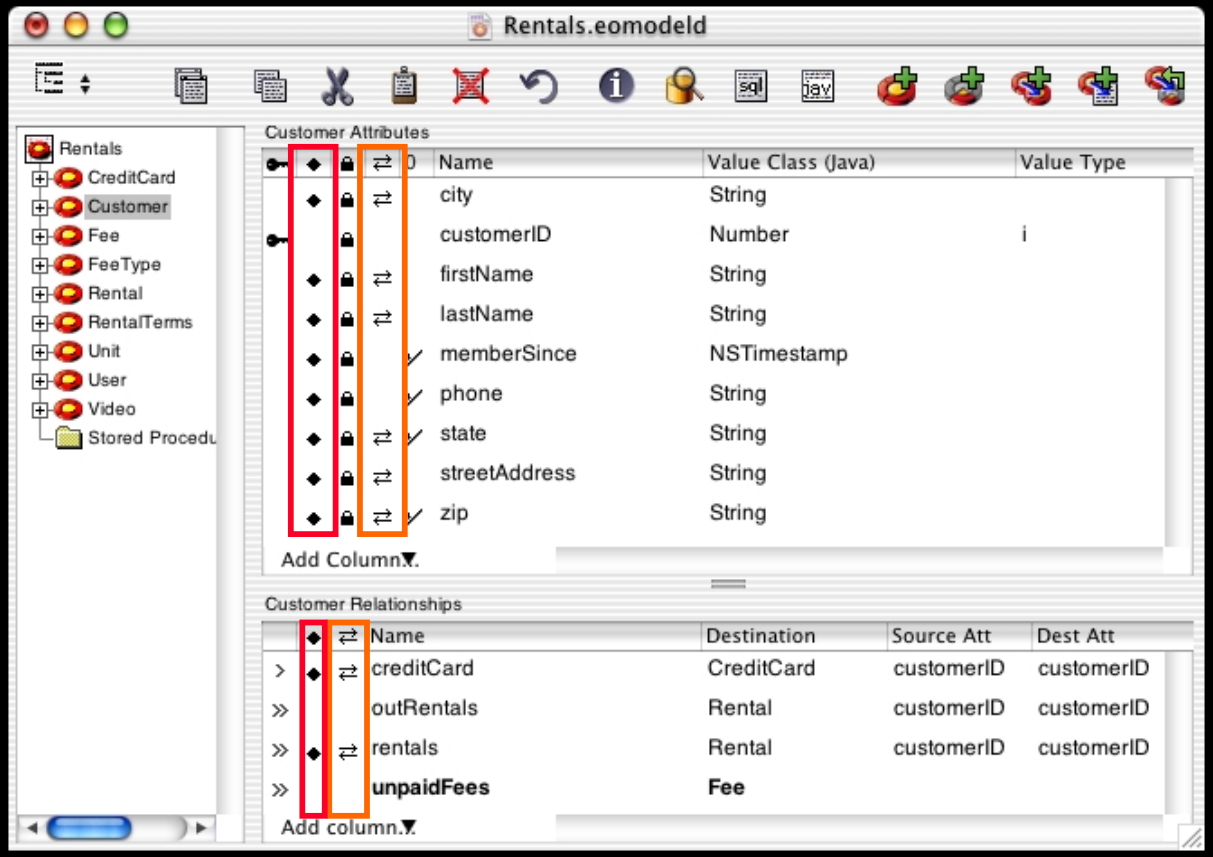
Rentals
- CreditCard
- Customer
- Fee
- FeeType
- Rental
- RentalTerms
- Unit
- User
- Video
- Stored Procedu

Name	Table	Class Name	Client-Side Class Name
CreditCard	CREDIT_CARD	example.common.CreditCard	example.common.CreditCard
Customer	CUSTOMER	example.server.Customer	example.client.Customer
Fee	FEE	example.common.Fee	example.common.Fee
FeeType	FEE_TYPE	example.common.FeeType	example.common.FeeType
Rental	RENTAL	example.common.Rental	example.common.Rental
RentalTerms	RENTAL_TERMS	example.common.RentalTerms	example.common.RentalTerms
Unit	UNIT	example.common.Unit	example.common.Unit
User	RENTALS_USER	example.server.User	example.client.User
Video	VIDEO	example.common.Video	example.common.Video

Add Column

Business Logic Partitioning

- Different server side and client side properties



Business Logic Partitioning

- Business logic code separation—Use API on EOEnterpriseObject

```
public Object invokeRemoteMethod (
    EOEditingContext editingContext ,
    String methodName ,
    Class[] argumentTypes ,
    Object[] arguments );

public void awakeFromClientUpdate (
    EOEditingContext editingContext );

public void prepareValuesForClient ();
```

Business Logic Partitioning

- Naming convention: “clientSideRequest” for unrestricted invocations
- Class design recommendation
  - Common superclass with shared functionality and abstract method declarations for partitioned logic
  - Client subclass with remote method invocations
  - Server subclass with concrete method implementations

Business Logic Partitioning

```
package example.common;

import com.webobjects.eocontrol.*;

public abstract class Secret extends EOGenericRecord {

    // shared functionality
    // ...

    public abstract String secretValue ();
}
```

Business Logic Partitioning

```
package example.client;

import com.webobjects.eocontrol.*;

public class Secret extends example.common.Secret {

    public String secretValue () {
        return (String) invokeRemoteMethod(
            “clientSideRequestSecretValue”,
            null, null);
    }
}
```

Business Logic Partitioning

```
package example.server;

import com.webobjects.eocontrol.*;

public class Secret extends example.common.Secret {

    public String secretValue () {
        return “secret”;
    }

    public String clientSideRequestSecretValue() {
        return secretValue();
    }
}
```

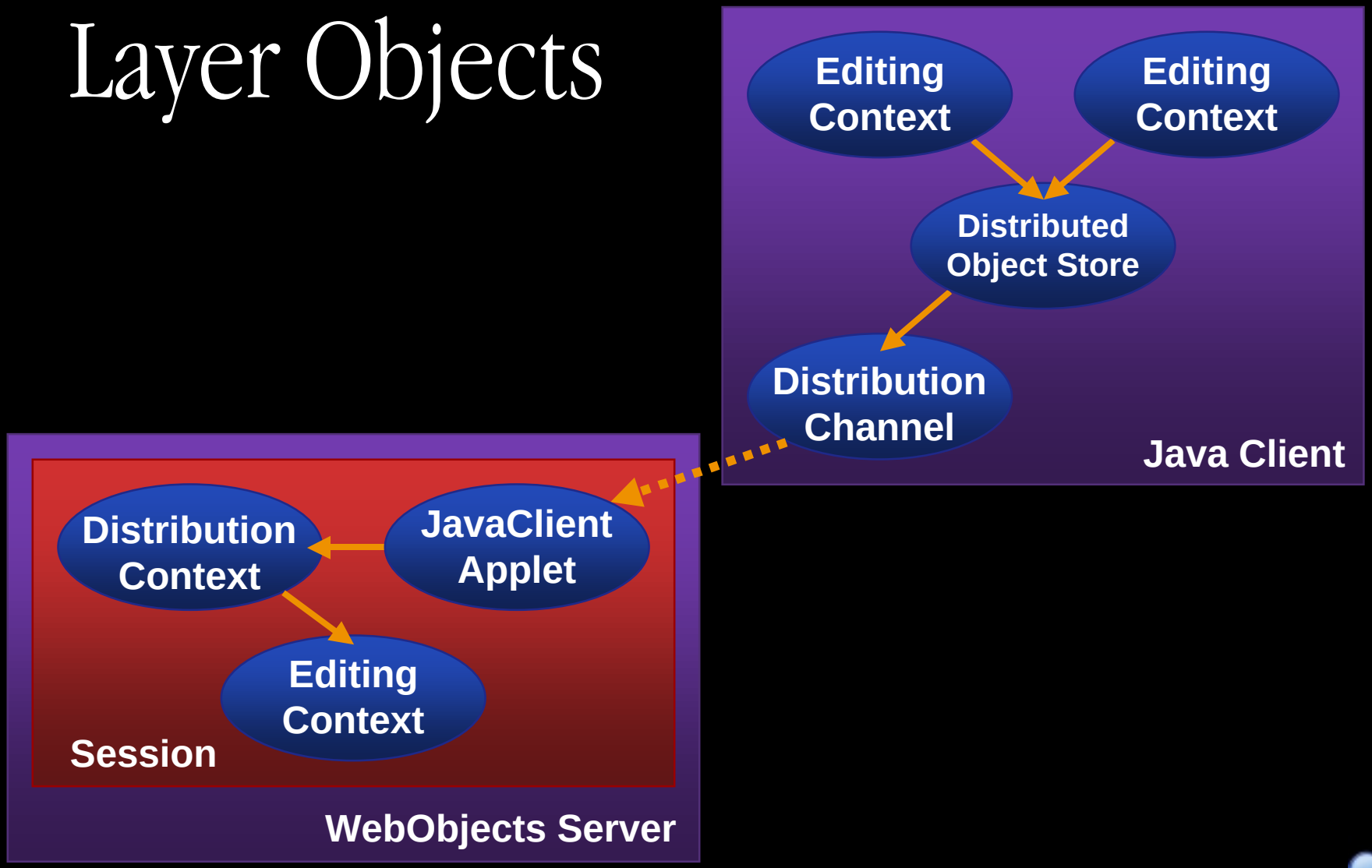
Agenda

- Distribution Layer
  - Business Logic Partitioning
  - Distribution Layer Objects
  - Remote Method Invocations
  - Distribution Channels
  - Delegates
- Deployment
- Direct to Java Client—Controller Factory

Distribution Layer Objects

- Server side
  - WOJavaClientApplet
  - EODistributionContext
- Client side
  - EODistributedObjectStore
  - EODistributionChannel (EOHTTPChannel)

Distribution Layer Objects



Agenda

- Distribution Layer
  - Business Logic Partitioning
  - Distribution Layer Objects
  - Remote Method Invocations
  - Distribution Channels
  - Delegates
- Deployment
- Direct to Java Client—Controller Factory

Remote Method Invocations

- EODistributedObjectStore is root object store
- Remote method invocation API

```
public Object invokeRemoteMethodWithKeyPath (
    EOEditingContext editingContext ,
    String keyPath ,
    String methodName ,
    Class[] argumentTypes ,
    Object[] arguments ,
    boolean pushChanges );
```

Remote Method Invocations

- Key path
  - Null: Distribution context itself
  - Empty String: Invoke on distribution context targets
    - Use distribution context notification to register additional targets

      EORemoteMethodReceiverNeededNotification
  - Other: Object relative to distribution context
    - Example: “session”
- Naming convention: “clientSideRequest”

Agenda

- Distribution Layer
  - Business Logic Partitioning
  - Distribution Layer Objects
  - Remote Method Invocations
  - Distribution Channels
  - Delegates
- Deployment
- Direct to Java Client: Controller Factory

Distribution Channels

- Distribution channels control the data transfer between client and server
  - Abstract superclass: EODistributionChannel
  - Concrete subclass: EOHTTPChannel
  - Custom subclasses enabled, but complicated
- Distribution channel class name is a standard client argument, specify in WOJavaClientApplet binding

Distribution Channels

- EOHTTPChannel— Subclass to create different sockets (for example for using SSL)

```
public Socket createSocket (
    String protocol ,
    String hostName ,
    int portNumber ) throws IOException;
```

Demo

Connecting through SSL

Norbert Schatz
WebObjects Engineer

Agenda

- Distribution Layer
  - Business Logic Partitioning
  - Distribution Layer Objects
  - Remote Method Invocations
  - Distribution Channels
  - Delegates
- Deployment
- Direct to Java Client: Controller Factory

Delegates

- EODistributionContext delegate
  - Encrypt/decrypt communication
  - Control access to business object fetching
  - Restrict remote method invocations
- EODistributionChannel delegate
  - Encrypt/decrypt communication
  - Handle server or I/O exceptions
    - Example: Session time outs

Delegates

- Setting delegates
  - Timing is important
  - EODistributionContext: Notification

    EODistributionContextInstantiatedNotification

  - EODistributionChannel: EOApplication subclass constructor

Agenda

- Distribution Layer
- Deployment
  - System Requirements
  - Deployment Options
  - Session Timeouts
  - Server Instances
- Direct to Java Client: Controller Factory

System Requirements

- Server: Standard WebObjects application
- Client: JDK 1.3 (contains Swing)

  Platforms:

  - Mac OS X: Integrated into OS
  - Windows: Download from Sun's web site
  - Mac OS 9: No JDK 1.3 support

Agenda

- Distribution Layer
- Deployment
  - System Requirements
  - Deployment Options
  - Session Timeouts
  - Server Instances
- Direct to Java Client: Controller Factory

Deployment Options

- Application: Create launch program
  - Mac OS X: Project Builder, MRJAppBuilder
  - Windows: Shortcut to launch script
- Applet
  - Mac OS X: IE, using Java VM in OS
  - Windows: IE/Netscape: Java plug-in required
    - Download from Sun's web site
  - Java Client uses Java Script to detect configuration

Demo

Creating Launch Program

Norbert Schatz
WebObjects Engineer

Deployment Options

	Application	Applet
Installation, Upgrade		●
User Experience	●	
Performance	●	
Isolation	●	
Security Restrictions	●	●
VM Parameters	●	

Deployment Options

- Good compromise: Use special class loaders for applications
  - Preinstall base system
    - JDK 1.3
    - EOF client classes
  - Download custom classes
    - Business logic
    - Custom controller classes
    - Custom JFC/Swing components

Agenda

- Distribution Layer
- Deployment
  - System Requirements
  - Deployment Options
  - Session Timeouts
  - Server Instances
- Direct to Java Client: Controller Factory

Session Timeouts

- Java Clients use standard WOSessions
  - Sessions time out
  - Desktop application users do not expect timeouts
- Java Client applications terminate sessions explicitly when user quits
- Use a large timeout (24 hours)

Agenda

- Distribution Layer
- Deployment
  - System Requirements
  - Deployment Options
  - Session Timeouts
  - Server Instances
- Direct to Java Client: Controller Factory

Server Instances

- Use multiple server instances for many clients, like you do for HTML based WebObjects applications
- Java Client Requests
  - High load at startup
  - Small packages afterwards, much smaller than HTML pages
  - Attention: Tasks performed in Intranets often need more data than HTML based applications for the Internet

Agenda

- Distribution Layer
- Deployment
- Direct to Java Client: Controller Factory

Controller Factory

- EOControllerFactory
  - Object in the center of Direct to Java Client
  - Requests controller hierarchies from server's rule system
  - Instantiates controller hierarchies from XML
- Useful for applications which are not completely generated dynamically
  - Mix dynamically generated user interfaces and static interface files
  - Provides APIs to retrieve configuration information from rule system

Controller Factory

- API to create controllers/windows

```
public NSArray selectWithEntityName (
    String entityName ,
    boolean allowsMultiSelection ,
    boolean allowsInserting );

public NSArray openGlobalIDsWithEntityName (
    String entityName ,
    NSArray globalIDs );

public EOController insertWithEntityName (
    String entityName );
```

Controller Factory

- API to create “primitive” controllers

```
public EOController queryControllerWithEntity (
    String entityName );

public EOController listControllerWithEntity (
    String entityName );

public EOController formControllerWithEntity (
    String entityName );

...
```

Demo

Dynamically Configured Subcontrollers

Norbert Schatz
WebObjects Engineer

Controller Factory

- Example: User customized table columns in an interface file based window
  - Write a rule to return the list properties from the configuration store
  - Place subcontroller area in interface file
  - Add subcontroller programatically using factory API

    listControllerWithEntity (configurationEntity);

  - Implement a way for users to configure display properties, store configuration

Summary

- Java Client is prepared for many advanced tasks
- Most important objects
  - Server side: EODistributionContext
  - Client side: EODistributedObjectStore
- Distribution channels customizable: SSL
- Multiple deployment options available—you choose
- Controller factory has very useful functionality

WebObjects Lab

- Located downstairs in Room K
- Lab hours
  - Monday 12:00pm–6:00pm
  - Tuesday 10:30am–6:00pm
  - Wednesday 10:30am–6:00pm
  - Thursday 10:30am–6:00pm
  - Friday 10:30am–3:30pm

☑ Vote for WebObjects as Best App Server

http://developer.apple.com/webobjects

Who to Contact

Toni Trujillo Vian
Director, WebObjects Engineering
webobjects@apple.com

Bob Fraser
WebObjects Product Manager
webobjects@apple.com

iServices Consulting, Integration, Training & Certification
(800) 848-6398
iservices@apple.com

For more information:
www.apple.com/webobjects
developer.apple.com/webobjects
www.apple.com/iservices
www.info.apple.com/webobjects

Subscribe to:
Webobjects-announce@apple.com

Q&A

Toni Trujillo Vian
Director, WebObjects Engineering
webobjects@apple.com

http://developer.apple.com/events/urls.html

WWDC
Worldwide Developers Conference 2001

Client Side Arguments

- Information client needs
  - Application class
  - Initial interface controller class
  - Distribution channel class
  - Direct to Java Client: Controller factory class
  - Applets: Height, width, archive, codebase, etc.

Client Side Arguments

- Arguments configured in one central place—WOJavaClientApplet
  - For both application and applet
- All bindings of WOJavaClientApplet become client arguments
- Add arbitrary bindings for custom arguments
- Arguments available on client side

```
EOApplication.sharedApplication().arguments()
```

Client Side Arguments

- Application startup
  - Connection information needs to be provided on command line
  - Command line arguments plus WOJavaClientApplet bindings form full set of arguments

WWDC
Worldwide Developers Conference 2001